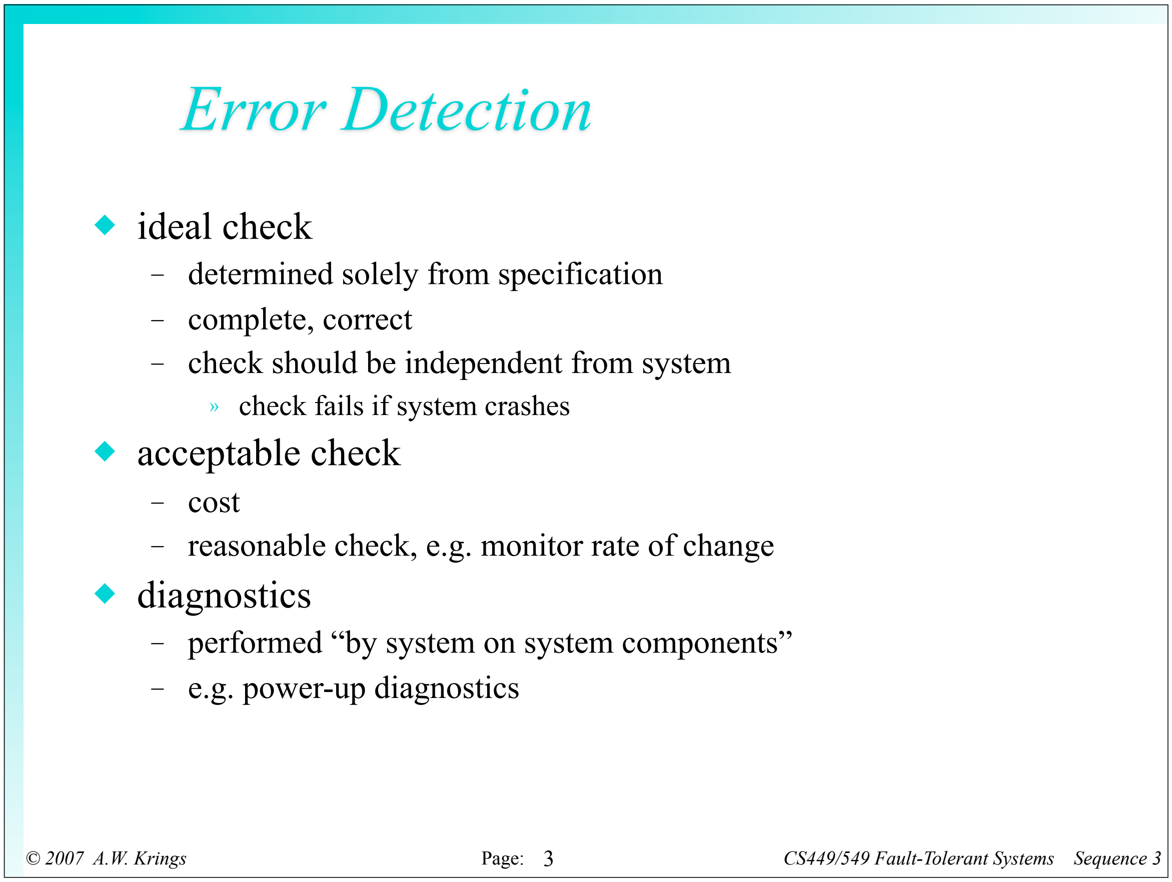
Redundancy

- Hardware redundancy
  - add extra hardware for detection or tolerating faults
- Software redundancy
  - add extra software for detection and possibly tolerating faults
- Information redundancy
  - extra information, i.e. codes
- Time redundancy
  - extra time for performing tasks for fault tolerance

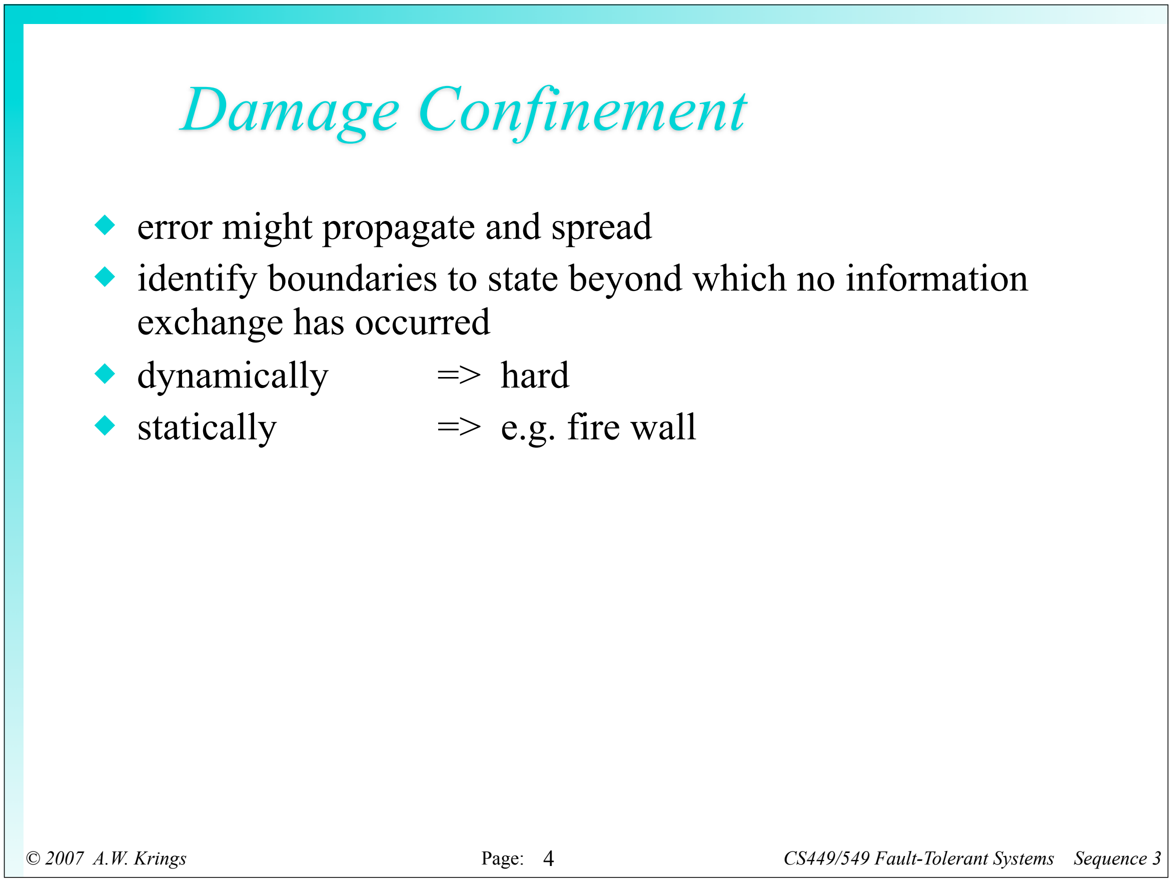
Fault Tolerance

- Error Detection
- Damage Confinement
- Error Recovery
- Fault Treatment



Error Detection

- ideal check
  - determined solely from specification
  - complete, correct
  - check should be independent from system
    - check fails if system crashes
- acceptable check
  - cost
  - reasonable check, e.g. monitor rate of change
- diagnostics
  - performed “by system on system components”
  - e.g. power-up diagnostics



Damage Confinement

- error might propagate and spread
- identify boundaries to state beyond which no information exchange has occurred
- dynamically => hard
- statically => e.g. fire wall

Error Recovery

- backward recovery
  - state is restored to an earlier state
    - requires checkpoints
  - most frequently used
  - recovery overhead
- forward recovery
  - try to make state error-free
  - need accurate assessment of damage
  - highly application-dependent

Fault Treatment

- if transient fault: restart system, go to error-free state
- system repair
  - on-line, no manual intervention, (automatic)
  - dynamic system reconfiguration
  - spare (hot or cold)

Fault Coverage

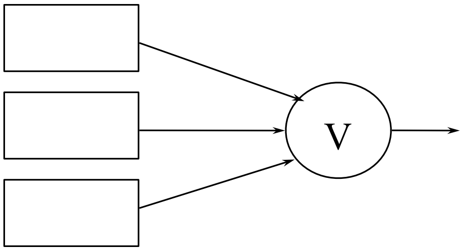
- measure of system’s ability to perform:
  - fault detection
  - fault location
  - fault containment
  - (and/or fault recovery)
- C = P(fault recovery | fault existence),
- Note:
  - recovery implies that the system as a whole is operational
  - this does not imply that a “repair” occurred
  - e.g. duplex system with benign fault can recover to continue operation on one non-faulty processor

Hardware Redundancy

- Passive (static)
  - uses fault masking to hide occurrence of fault
  - no action from the system is required
  - e.g. voting
- Active (dynamic)
  - uses comparison for detection and/or diagnoses
  - remove faulty hardware from system => reconfiguration
- Hybrid
  - combine both approaches
  - masking until diagnostic complete
  - expensive, but better to achieve higher reliability

Passive Hardware Redundancy

- N-Modular Redundancy (NMR)
  - N independent modules replicate the same function
    - parallelism
  - results are voted on
  - requirements: N >= 3
- TMR (Triple Modular Redundancy)



Voter:

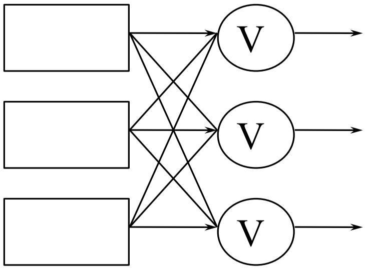
• is single point of failure.

• could be very simple,

• but *<u>who guards the guard</u>*?

Who guards the guards?

- Replicate voters



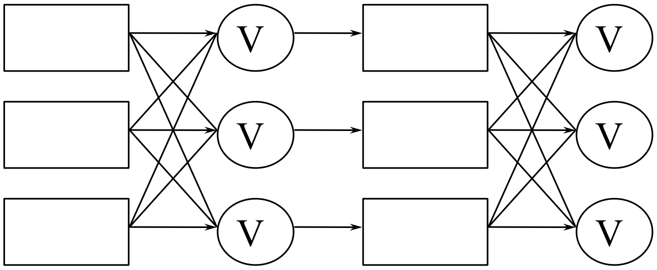
Restoring Organ:

since it produces 3 correct outputs even if one input is faulty.

eliminate single point of failure

Who guards the guards?

- Multistage TMR with replicate voters

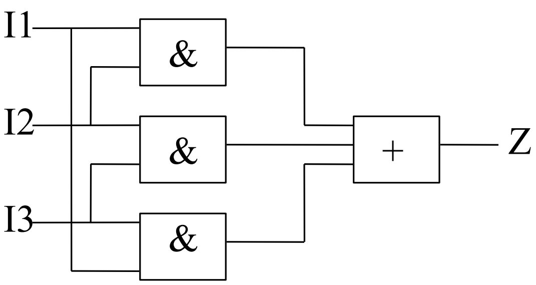


Voting

- if inputs are independent, the NMR can mask up to

$$\left\lfloor \frac{(N-1)}{2} \right\rfloor \quad \text{Faults}$$

- e.g. 1 bit majority voter (3 AND gates ORed)



Z=1 if 2 of 3 inputs are 1

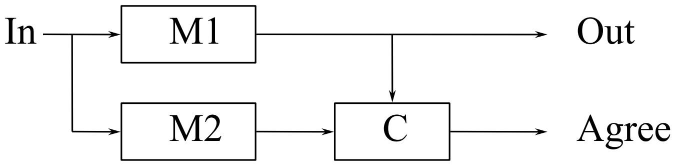
Z=0 if 2 of 3 inputs are 0

Flux Summing

- Inherent property of closed loop control system
- If one module becomes faulty, remaining modules compensate automatically.

Active Hardware Redundancy

- Duplicate and Compare



  - can only detect, but NOT diagnose
    - i.e. fault detection, no fault-tolerance
  - may order shutdown
  - comparator is single point of failure
    - simple implementation: 2 input XOR for single bit compare

Active Hardware Redundancy

Johnson 1989

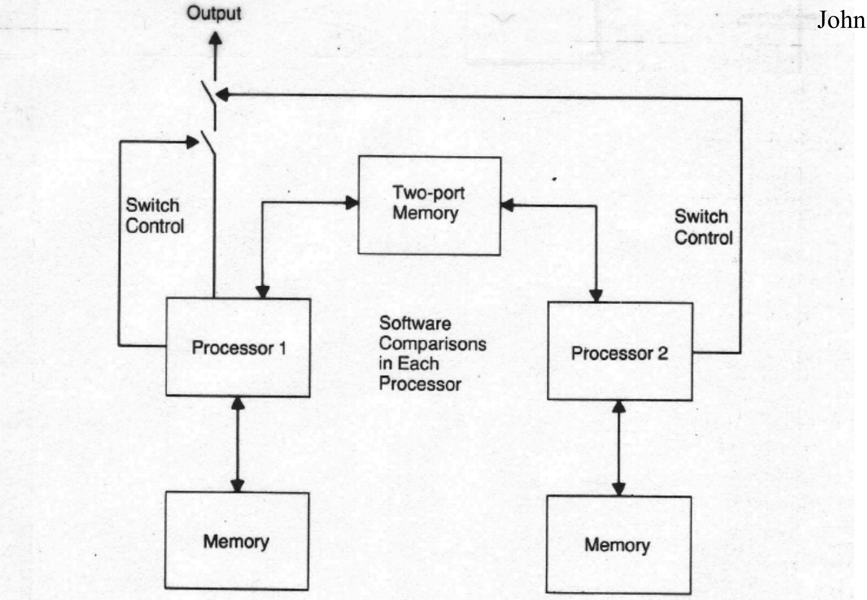


Fig. 3.13 The necessary comparisons in duplication with comparison can be implemented in software. Both processors must agree that results match before an output is generated.

Active Hardware Redundancy

- Stand-by-sparing
  - only one module is driving outputs
  - other modules are
    - idle => hot spares
    - shut down => cold spares
  - error detection => switch to a new module
  - hot spares
    - no power-up delays
    - power consumption
  - cold spares
    - opposite of hot spares

Johnson 1989

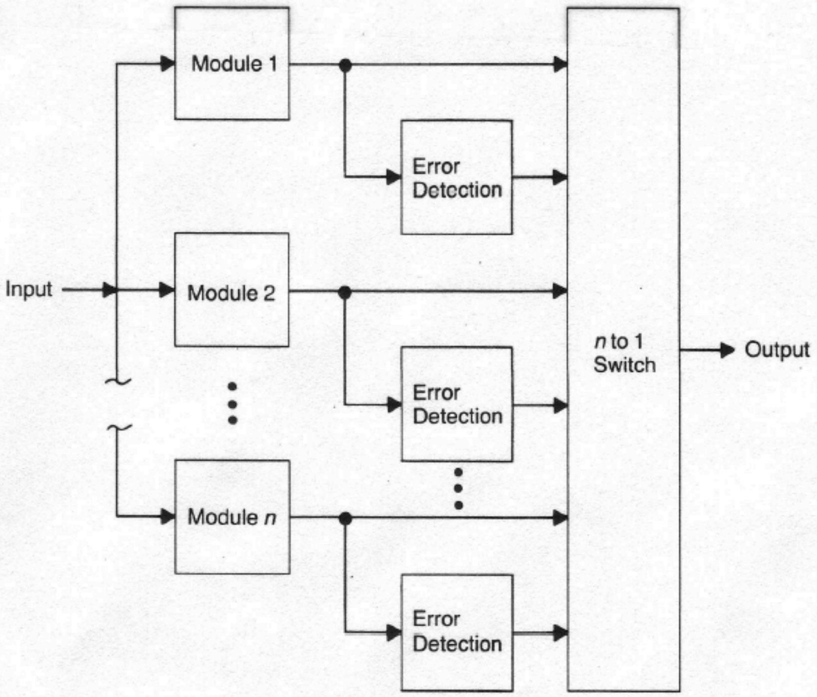


Fig. 3.14 In standby sparing, one of n modules is used to provide the system's output, and the remaining $n - 1$ modules serve as spares. Error detection techniques identify faulty modules so that a fault-free module is always selected to provide the system's output.

Active Hardware Redundancy

- Pair and Spare
  - duplication combined with compare & spare
  - 2 modules are always on-line
  - 2-of-N switch
  - pairs are often combined

Johnson 1989

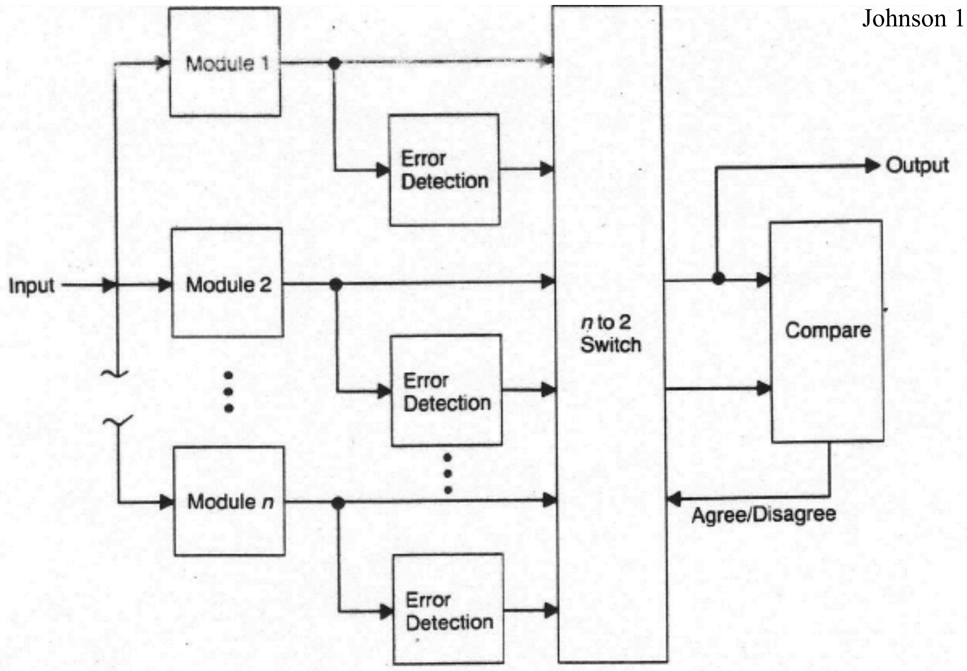


Fig. 3.15 The pair-and-a-spare technique combines duplication with comparison and standby sparing. Two modules are always online and compared, and any spare can replace either of the online modules.

Hybrid Hardware Redundancy

- NMR with spares
  - N active + S spare modules (off-line)
  - voting and comparison
  - replace erroneous module from spare pool
  - maintains N constant
  - uses N-of-(N+S) switch
- example: 2 faults at 2 different times
  - hybrid solution => N = 4
  - passive solution => N = 5

$$\left\lfloor \frac{(N-1)}{2} \right\rfloor$$

Johnson 1989

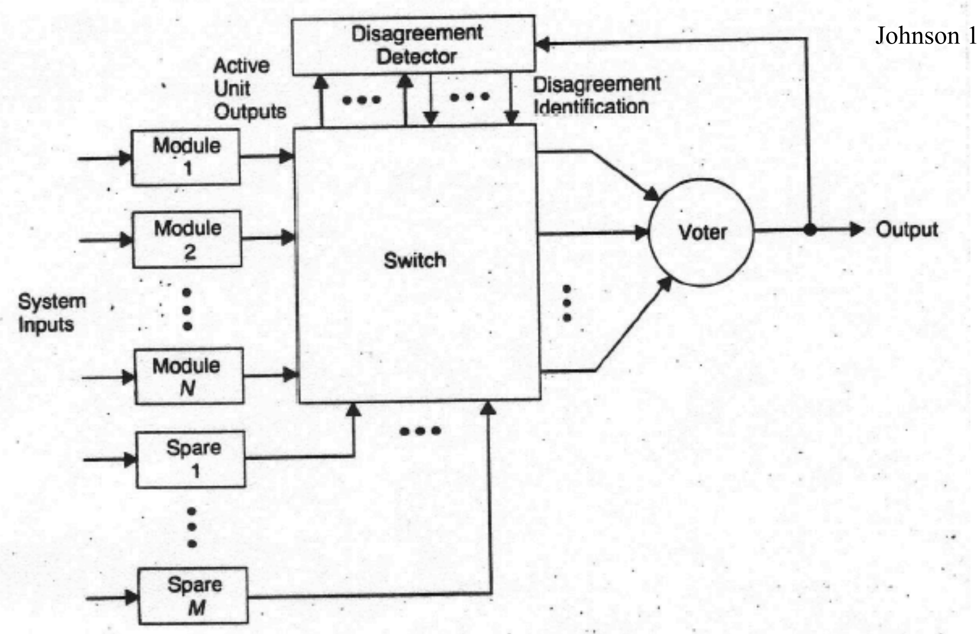


Fig. 3.16 *N*-modular redundancy with spares combines NMR and standby sparing. The voted output is used to identify faulty modules, which are then replaced with spares.

Hybrid Hardware Redundancy

- Self-purging NMR (Joh89 Fig 3.17)
  - all modules are active
  - exclude modules on error detection
    - vote & compare
  - N will decrease with faults

Johnson 1989

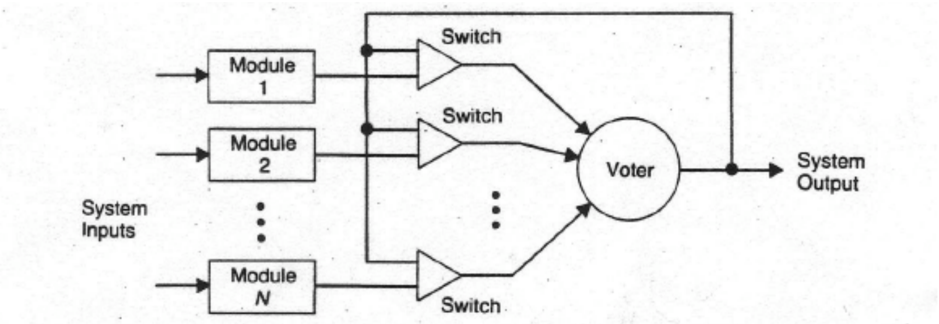


Fig. 3.17 **Self-purging redundancy uses the system output to remove modules whose output disagrees with the system output.** (From [Losq, 1976] © 1976 IEEE)

Hybrid Hardware Redundancy

- Triple-Duplex (Johnson 1989 Fig. 3.26, page 80)
  - redundant self checking
  - each node is really 2 modules + comparator
  - self-disable in event of error
  - “simulate” benign behavior
  - triple-triplex used in Boeing 777 primary flight computer
    - each triplex node employs 3 dissimilar processors

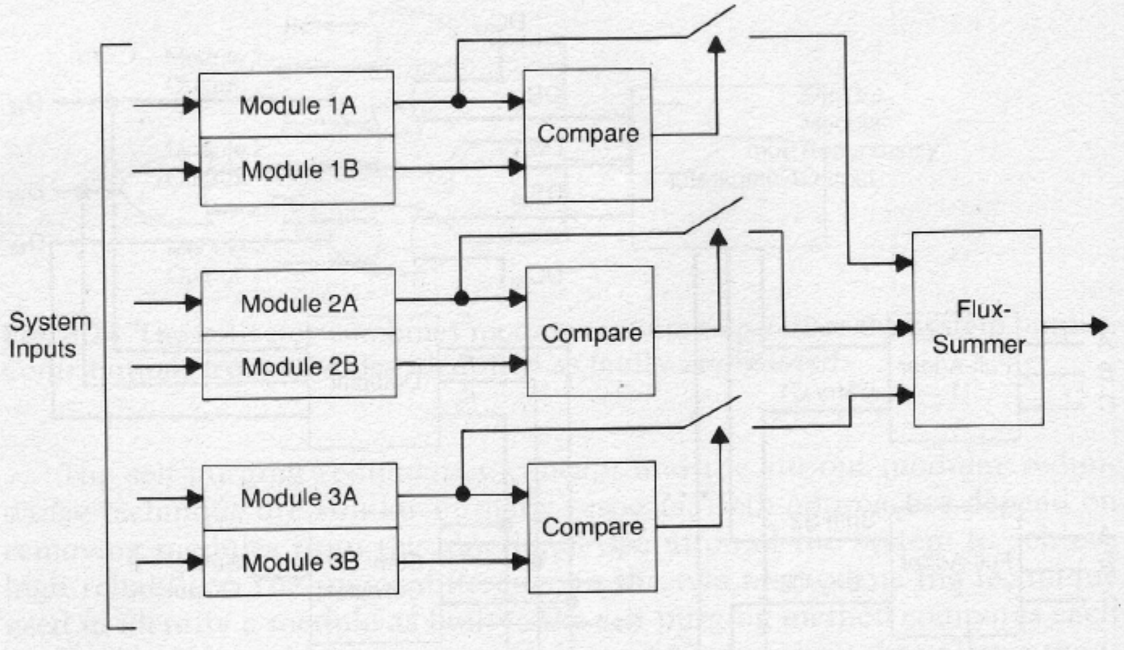


Fig. 3.26 The triple-duplex architecture uses duplication with comparison to detect faulty modules, and triplication is used to provide fault masking.